CSSE1001

Semester 1, 2015

Assignment 3 - Instant Messenger Topic Suggestion

1 Introduction

An instant messenger (commonly referred to as an 'IM') is a program that facilitates synchronous communication between two or more clients over a network.

There are many well-known implementations of IM systems that you may already be familiar with, for example:

`Facebook IM`	An IM system integrated into the facebook site.
`irssi`	A terminal-based IM for UNIX systems that uses the IRC communication protocol.
`Skype IM Service`	A stand-alone IM application using a propietary, closed source network infrastructure.

You may choose to take inspiration from some of the elements of these applications in your own assignments.

2 System Topology

One of the largest design decisions you are likely to face when constructing an instant messenger is how to structure communication between clients. There are many possible ways to structure an instant messaging service. Whilst there is no single 'correct' way to do this, some methods are more ideal than others. A few simplistic system topologies have been briefly outlined for you. Please note that this list is by no means exhaustive. You should not feel restricted by these suggestions.

2.1 Peer-To-Peer Using a Name Server

A *Name Server* is a program with an IP address and port number known by each messenger client. When one client wishes to begin a communication it requests the IP address and port number of the other client process from the Name Server, allowing a peer-to-peer connection to begin (see Figure 1).

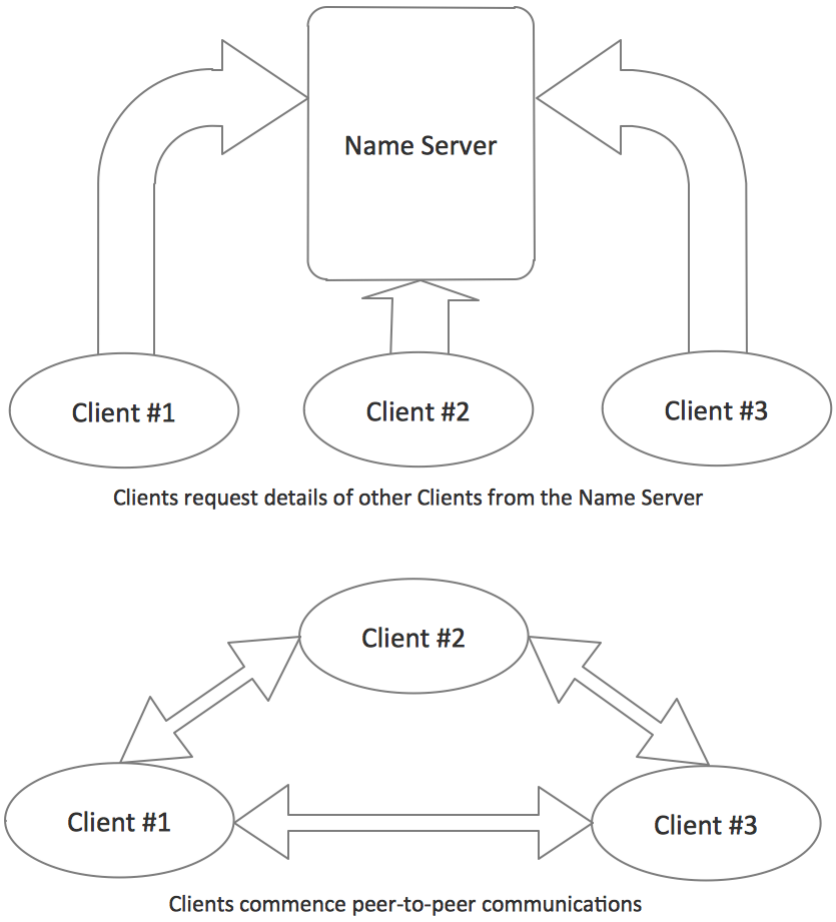

Figure 1: Simplified Peer-to-Peer network topology

2.2 Central Server Topology

A program that contains address information for all active clients acts as a 'postal service' for messages between clients. That is, if client A is communicating with client B, client A would send a message to the IM server and the IM server would consequently send the message to client B (see Figure 2). This configuration requires the client processes to register themselves with the IM server prior to commencing any other communications.

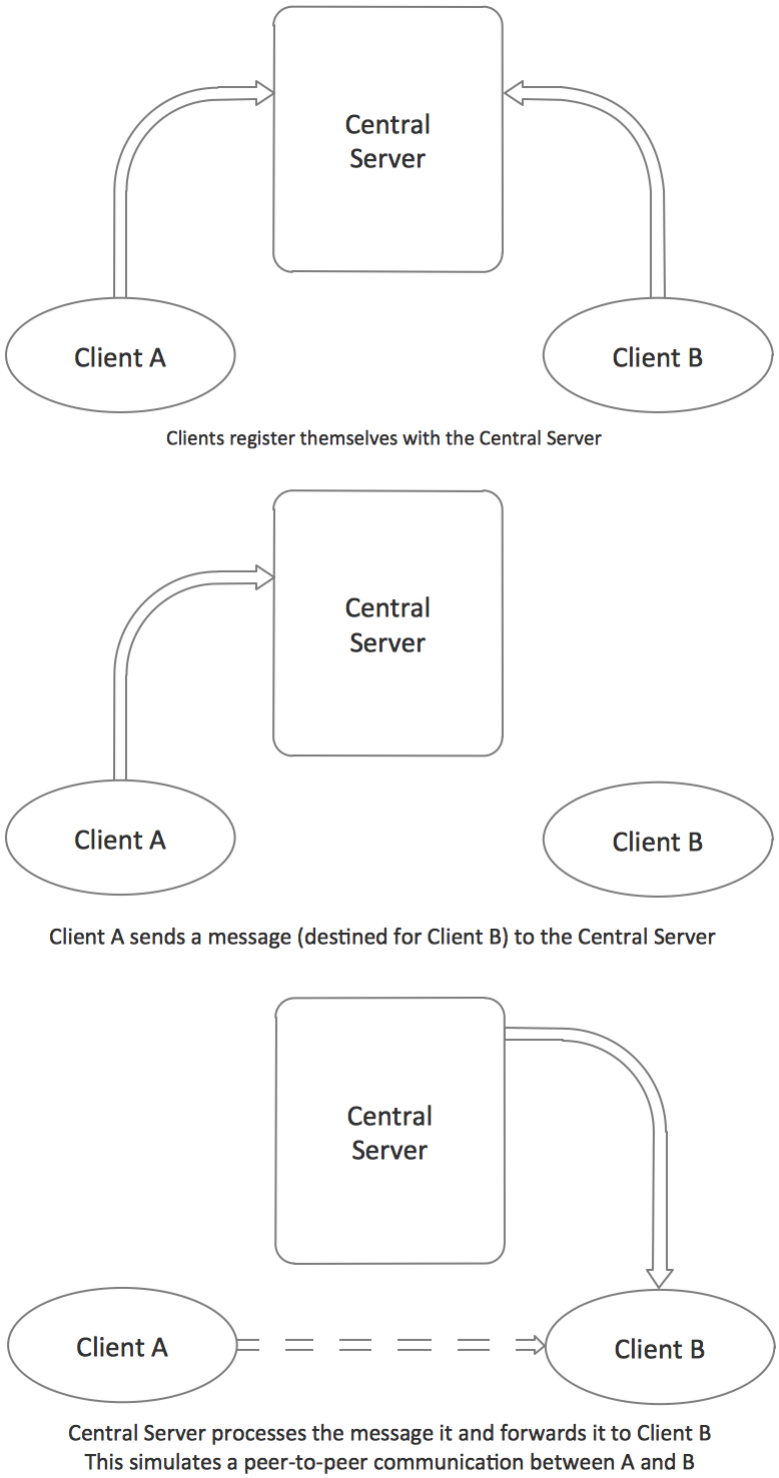

Figure 2: Simplified central server network topology

2.3 Publish/Subscribe Message Transport Topologies

Messages can be broadcast to a number of client processes simultaneously by using a publish/subscribe transport system such as MQTT (see `mqtt.org/` for protocol documentation).
This configuration involves a number of clients *subscribing* to a process and receiving all messages *published* to that same location by other clients (see Figure 3).

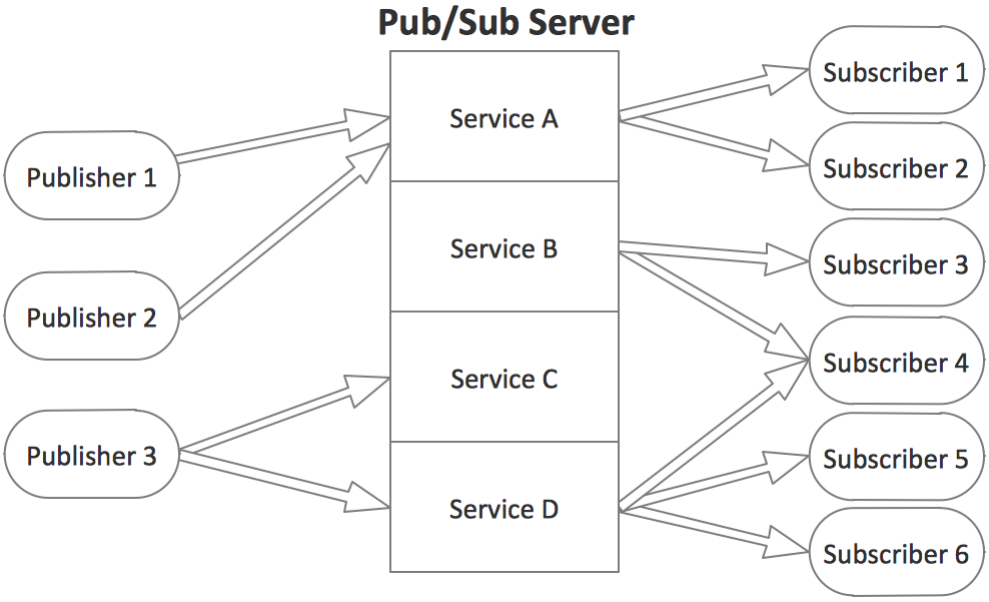

Figure 3: Simplified publish/subscribe system topology

3 Expecations

As is eluded to in the Assignment 3 task sheet, the more complexity you add to your assignment the higher your assignment's rank. Since the design decisions involved in this assignment are to be yours, a number of possible features for inclusion in your assignment have been included.

It should be noted that this list is neither exhaustive nor compulsory. You may choose to implement any number of these features as well as ones not included here.

3.1 Basic Features

- Some form of User Interface
  - Text interface (similar to the IRSSI client)
  - Graphical user interface (similar to the *Skype* interface)
- Capactiy to hold concurrent conversations
  - This may require knowledge of some concurrency programming techniques (see section 4.3)
- Client addressing system (i.e. some form of an address book)

3.2 Advanced Features

- Asynchronous communication between hosts
  - Messages sent to an unavailable host will be delivered once the host is next visible
- Multi-host conversations
  - Allow conversations between three or more different clients
- Persistent conversations between login sessions
  - Maintain a history of conversations so that a conversation is not lost once a conversation session is terminated
- Image and file transfer
  - This may require research of file transfer protocols such as *ftp* and *http*

4 Resources

You will need to use a number of resources in order to implement this assignment. Some possibly useful libraries include:

4.1 Networking Resources

socket.socket

Comes packaged as a part of the standard Python 3 distribution. There is extensive documentation and examples on how to use Python sockets online. One such resource can be seen at https://docs.python.org/3/library/socket.html.

paho-mqtt

`paho-mqtt` is an implementation of the MQTT protocol for Python. Installation guidlines can be found at https://pypi.python.org/pypi/paho-mqtt. Basic usage instructions can also be found here.

4.2 User Interface Resources

TkInter

`TkInter` will come installed as a part of the standard Python 3 distribution. This is the library used in CSSE1001 to teach Graphical User Interfaces. Guidance on using this resource can be found in the course notes as well as online.

PyQt4

PyQt4 is a popular GUI library written by Riverbank Computing Limited. It has a large community following and can be installed for Windows from the Riverbank website (www.riverbankcomputing.com/software/pyqt/download). OS X systems can install PyQt4 by installing homebrew (brew.sh) and running the following commands in a terminal window:

```
brew install sip --with-python3
brew install pyqt --with-python3
```

4.3 Concurrency Resources

One of the simplest methods of achieving concurrency is through the use of *multi-threading*.
Threads are a service provided by some operating systems to user applications that allow elements of a program to be run concurrently. Some implementations of instant messengers may require such services.

Thankfully, a threading library is included in the standard Python 3.4 distribution. This library, named `threading`, is well documented at locations such as the Python website: (https://docs.python.org/3/library/threading.html). Example implementations of multithreaded applications (such as the examples found at http://pymotw.com/2/threading/) are also easily accessible.